



LOGIKAS
CONECTANDO IDEAS

Unidad I “Toma de contacto”



This work by **Logikas** is licensed under a ***Creative Commons Reconocimiento-NoComercial-SinObraDerivada 3.0 Unported License***

Presentación:

Les damos la bienvenida al curso Hands On GWT, promocionado por Juglar.org e impulsado por la empresa Logikas. El nivel de este curso es introductorio, y pretende dotar a los alumnos de los conocimientos básicos necesarios para comenzar a desarrollar un proyecto con la plataforma GWT.

Un poco de Historia:

Desde la revolución que inició la tecnología AJAX han surgido diferentes librerías y toolkits basados en JavaScript, persiguiendo el objetivo común de eliminar las diferencias entre navegadores, y dotar al humilde entorno JavaScript de mayor potencia de desarrollo. JavaScript por sí solo no impone ningún tipo organización en el código (ni clases ni paquetes), no provee de ningún API de carga incremental de código ni impone restricciones de división entre código ejecutable y representación de la vista. Todo esto debe ser provisto por toolkits de desarrollo externos, como Prototype, jQuery, Dojo, MooTools, YUI y Ext JS. Aunque estas carencias parezcan desventajas de JavaScript, en realidad son su fortaleza, ya que no imponen rigidez en el desarrollo, y permiten estructuraciones de código muy diversas. Esto tiene sentido si pensamos que JavaScript es un lenguaje de “pegamento” en un entorno tan heterogéneo como la web, en donde la página/aplicación debe nutrirse de servicios de diferentes proveedores, cada uno con sus propias características.

¿En qué se diferencia GWT de los toolkits clásicos como Prototype y jQuery?

GWT es algo completamente diferente a los toolkits “clásicos” de JavaScript. GWT utiliza a Java como lenguaje de programación en lugar de JavaScript. Para ello, compila el código Java y emite una salida en JavaScript, que puede ser interpretada por los navegadores.

A diferencia de lo que sucede en los demás toolkits, en GWT las aplicaciones son del tipo Single Page, porque corren sobre una única página web durante todo su ciclo de vida.

¿Qué sentido tiene utilizar Java en el navegador cuando puedo utilizar JavaScript?

Java es quizás el lenguaje de propósito general que más herramientas de desarrollo tiene a su disposición. Entornos como Eclipse, NetBeans e IntelliJ, trabajan con Java como lengua madre, ofreciendo características de depuración y refactorización que no serían sencillas de reproducir en



un lenguaje como JavaScript. Los frameworks de prueba como JUnit, no tienen equivalentes tan poderosos en JavaScript. Por otra parte, la estructuración natural de código por medio de paquetes y clases, permite afrontar proyectos de gran envergadura, en los que participan decenas de desarrolladores.

Además de todo esto la característica de **Tipado Estático** de Java, abre la puerta para optimizaciones que serían imposibles en lenguajes de **Tipado Dinámico** como JavaScript. El compilador de GWT aplica lo que se conoce como Optimización Agresiva, que permite eliminar todo el código redundante de una aplicación, haciendo que el código generado sea mucho más óptimo que el que podría escribirse en JavaScript “a mano”.

Por último, el hecho de utilizar el mismo lenguaje tanto en cliente como en servidor, hace que el código sea más fácil de mantener, y permite que los desarrolladores, aunque

no poseen mucho conocimiento del entorno web, puedan generar aplicaciones al mismo nivel que un experto en JavaScript.

1) Instalación de Herramientas:

Para comenzar con el curso, y mantener una estructura similar de instalación en nuestras máquinas, lo más conveniente es que todos tengamos la misma jerarquía de directorios. Para ello crearemos un directorio denominado **handsongwt** en nuestro sistema de archivos. Dentro de este directorio depositaremos todos los recursos que desarrollemos durante el dictado del curso.

1.1) Instalación de Google Chrome:

Para este curso utilizaremos el navegador Google Chrome como navegador primario. Para instalarlo podemos descargar su instalador desde este enlace y luego correrlo.



1.2) Instalación de Eclipse:

La versión de Eclipse que instalaremos será Galileo, y la descargaremos del área de descargas de eclipse <http://eclipse.org/downloads/>

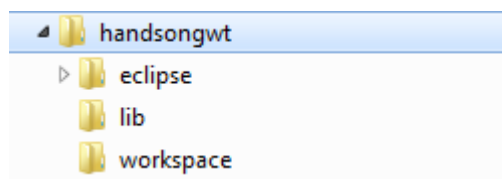


Eclipse IDE for Java Developers (92 MB)

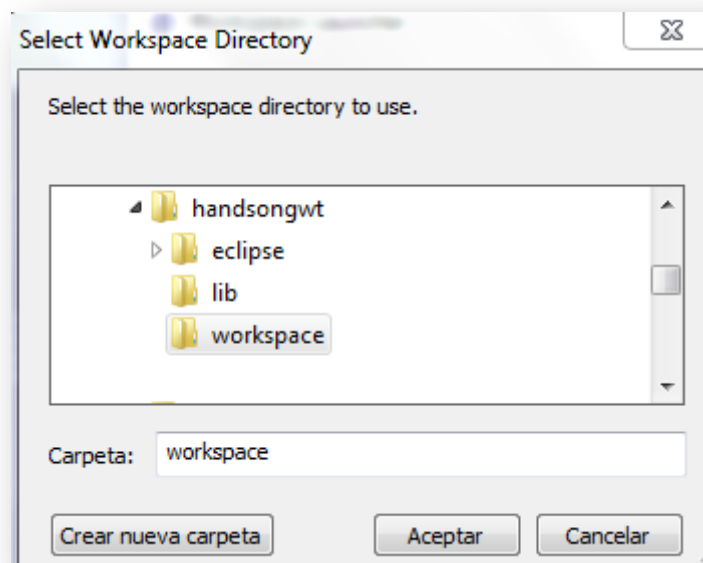
The essential tools for any Java developer, including a Java IDE, a CVS client, XML Editor and Mylyn. [More...](#)
Downloads: 605,266

Windows 32bit
Mac Carbon 32bit
Mac Cocoa 32bit 64bit
Linux 32bit 64bit

Una vez descargado el archivo, lo extraeremos dentro de nuestro directorio **handsongwt**. También crearemos un directorio **workspace** y un directorio **lib** al mismo nivel del directorio **eclipse** recientemente creado.

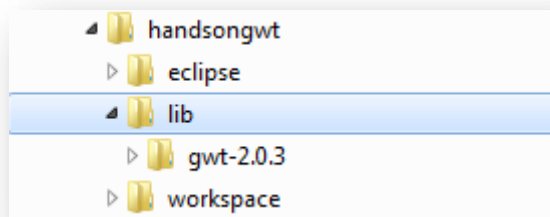


Comprobaremos que Eclipse fue correctamente instalado, corriendo el ejecutable eclipse (eclipse.exe en Windows) que se encuentra dentro del directorio **eclipse**. Eclipse nos pedirá la ruta del directorio **workspace** que deseamos utilizar, entonces seleccionaremos el directorio **handsongwt/workspace**.



1.3) Instalación del SDK de GWT:

El SDK de GWT es el conjunto de herramientas que nos permitirá desarrollar nuestras aplicaciones. El SDK contiene las librerías de cliente y servidor, el compilador, y el entorno virtual para desarrollo (Development Mode). Nosotros descargaremos la versión más reciente que es GWT 2.0.3 y se encuentra disponible en este enlace. Luego extraeremos el archivo descargado dentro la carpeta **handsongwt/lib** .

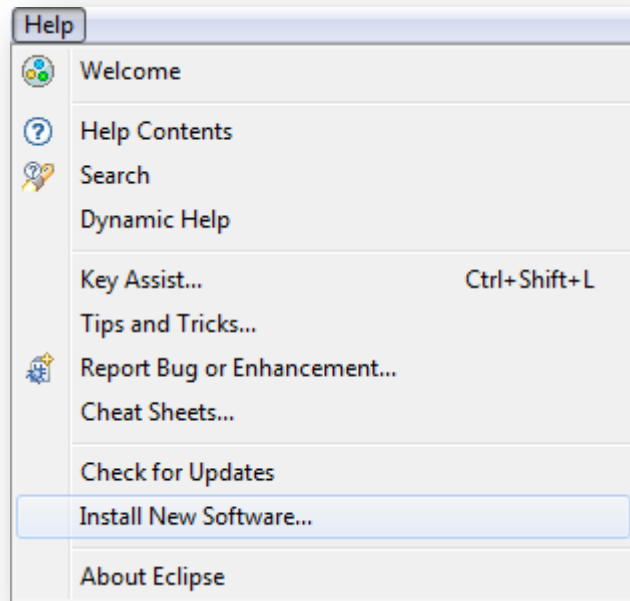


1.4) Instalación del Google Eclipse Plugin:

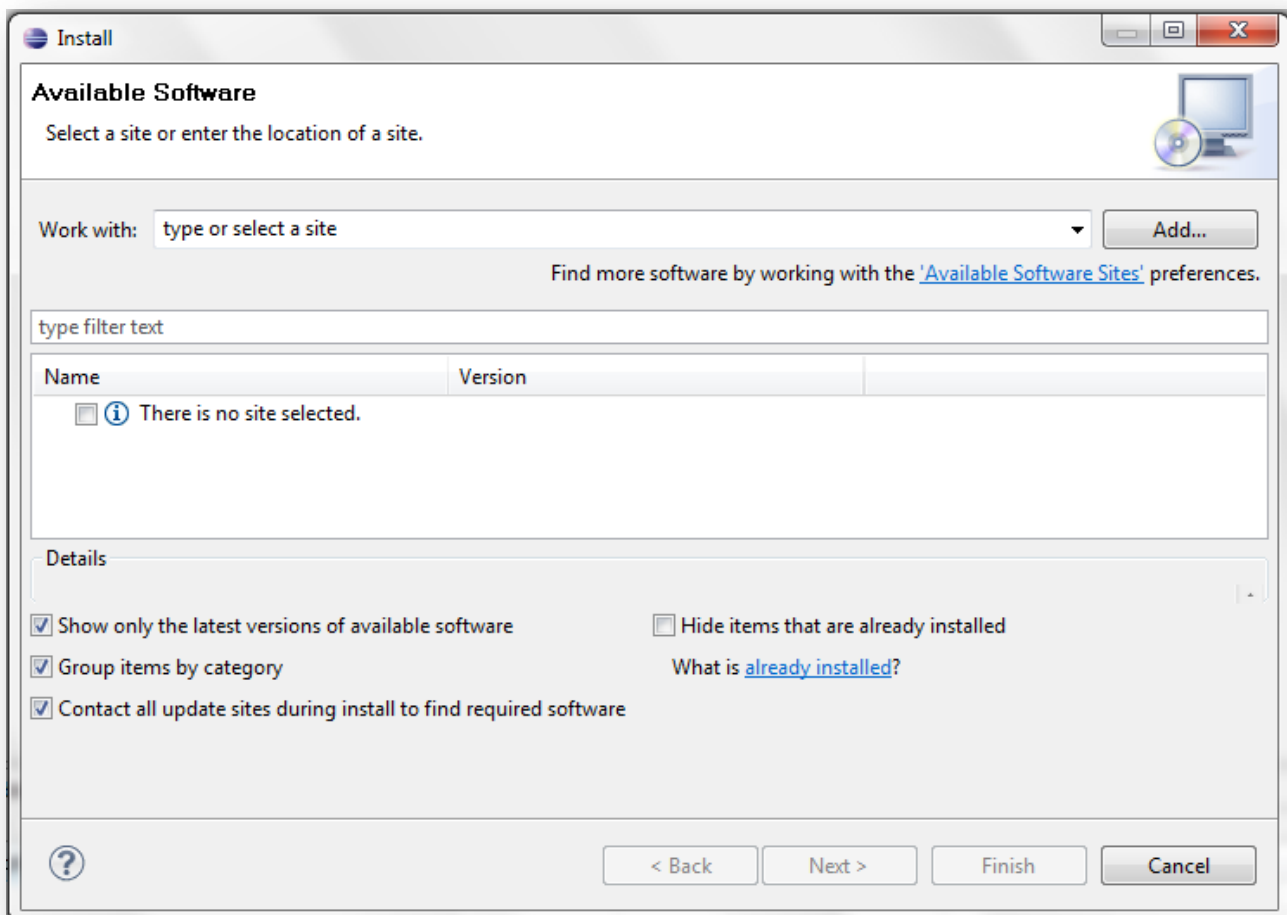
El **Google Eclipse Plugin** es el conjunto de extensiones que nos permitirá desarrollar los proyectos GWT dentro del entorno Eclipse. Para instalar el Plugin de Eclipse, utilizaremos el centro de actualizaciones automáticas, cuya URL de actualización es <http://dl.google.com/eclipse/plugin/3.5> (Si conoces cómo utilizar el centro de actualizaciones automáticas de Eclipse, puedes saltearte hasta la próxima sección).



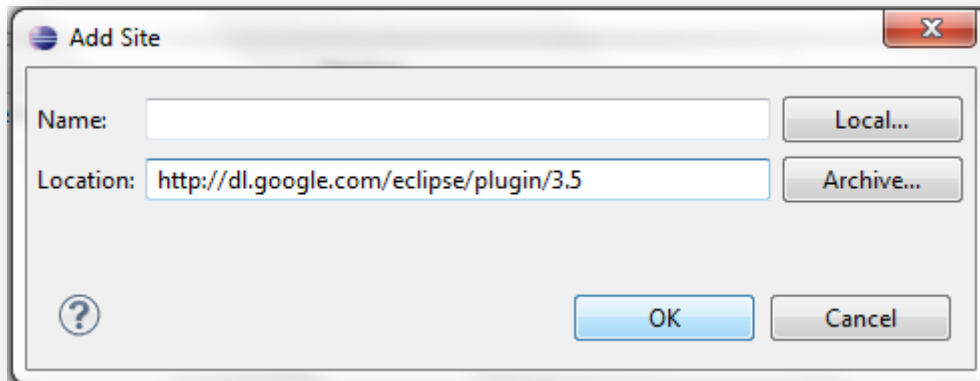
Desplegando el menú “Help” de Eclipse, haremos click sobre “Install New Software...”



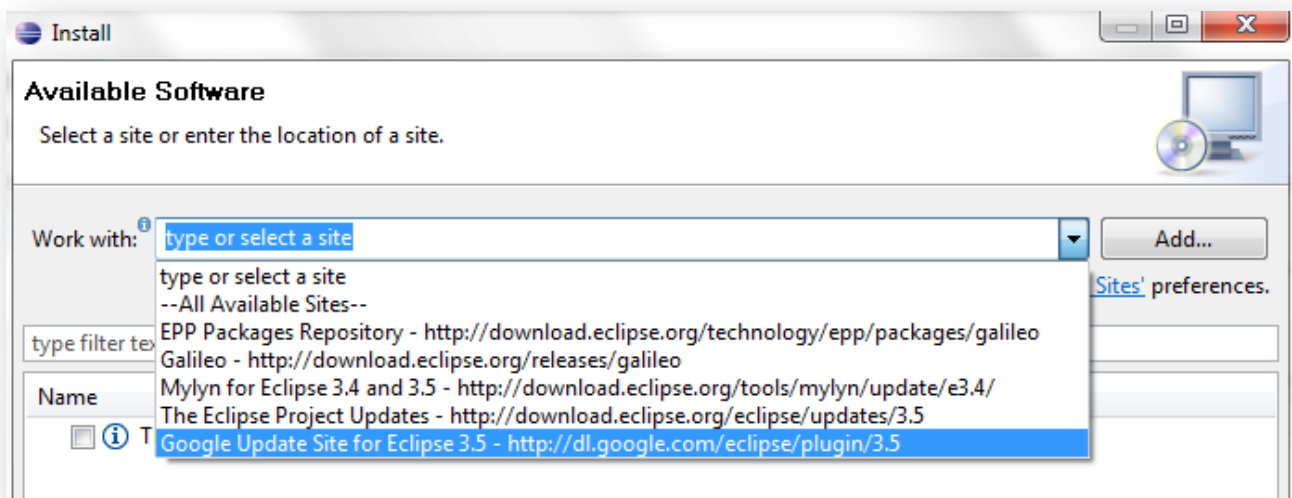
En el cuadro de diálogo “Install” haremos click sobre el botón “Add...”



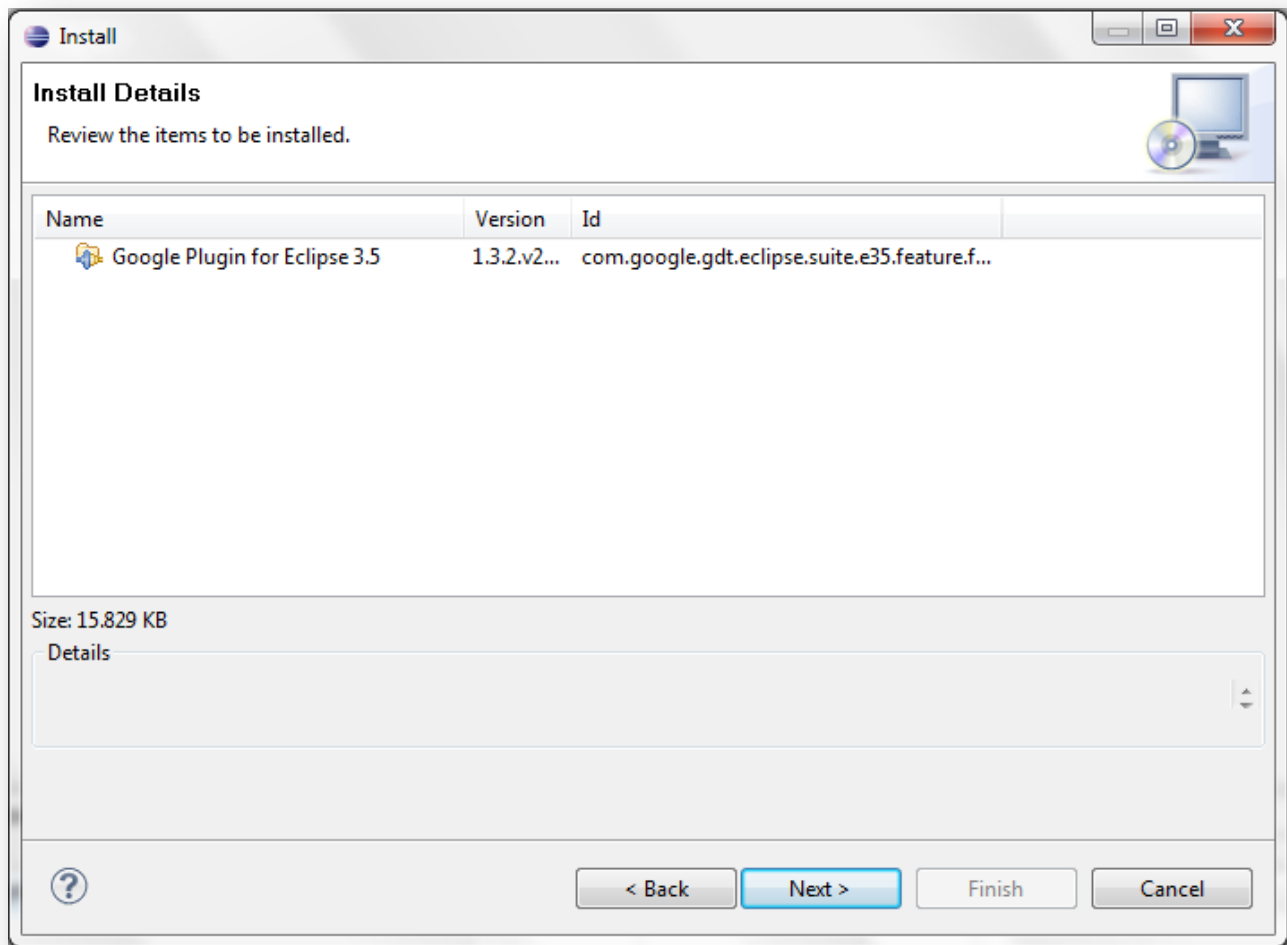
En la caja de texto “Location” escribimos la URL de actualización del plugin de Google, que es <http://dl.google.com/eclipse/plugin/3.5>, y haremos click sobre “Ok”.



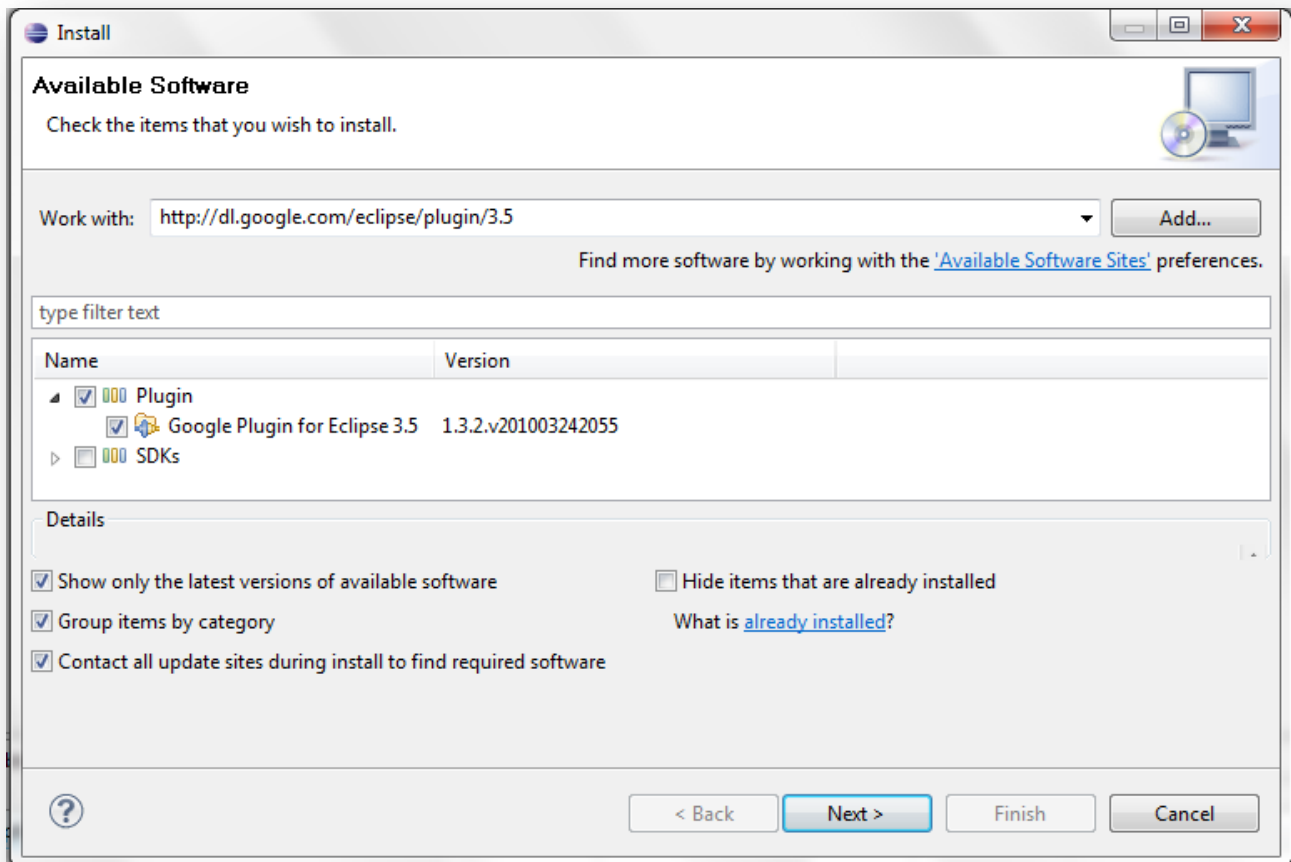
Sobre el cuadro de diálogo “Install”, seleccionaremos “Google Update Site for Eclipse 3.5” del cuadro desplegable “Work with”.



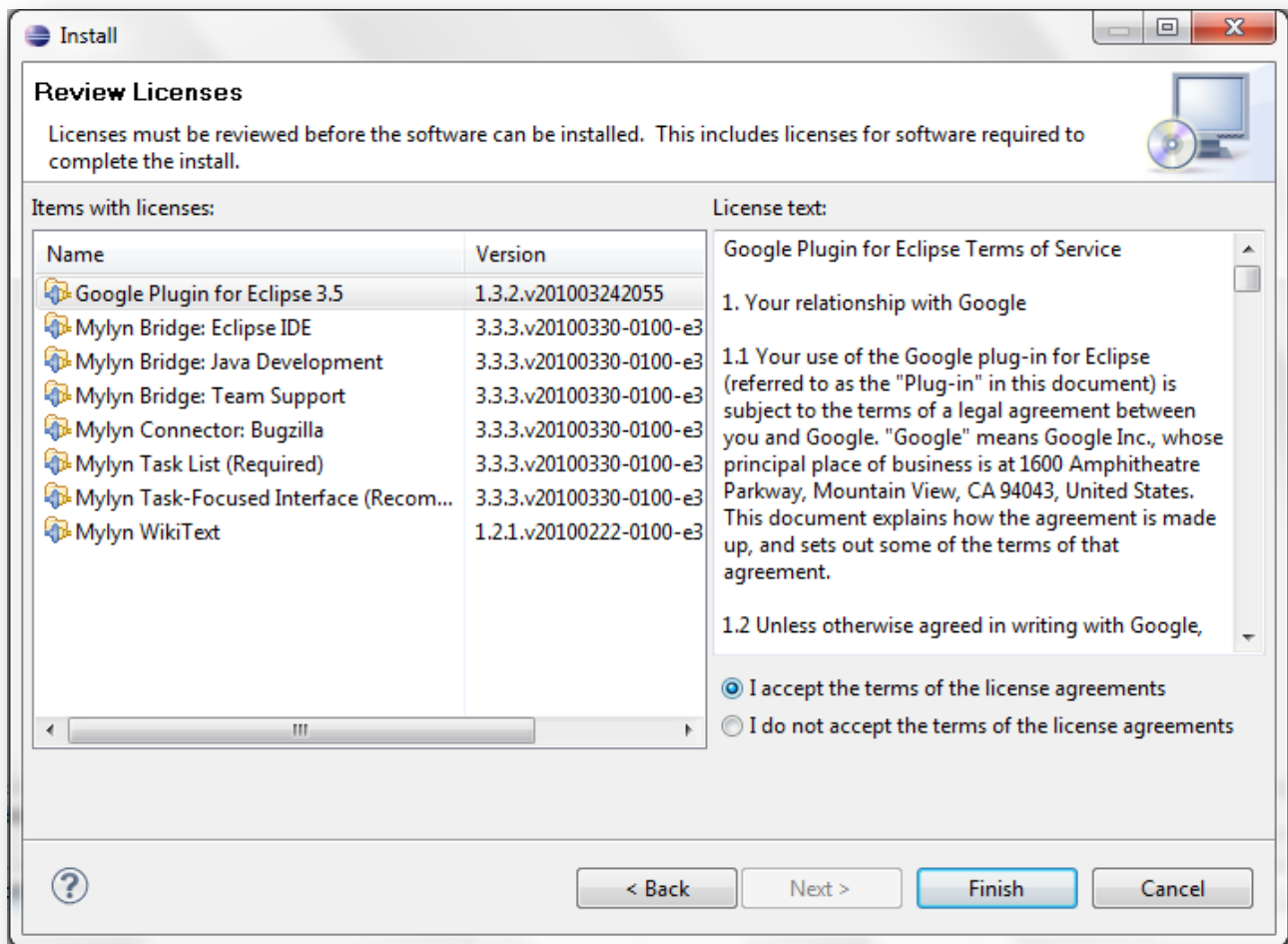
Esto nos mostrará una serie de paquetes a instalar. Nosotros seleccionaremos “Plugin” y haremos click sobre el botón “Next”.



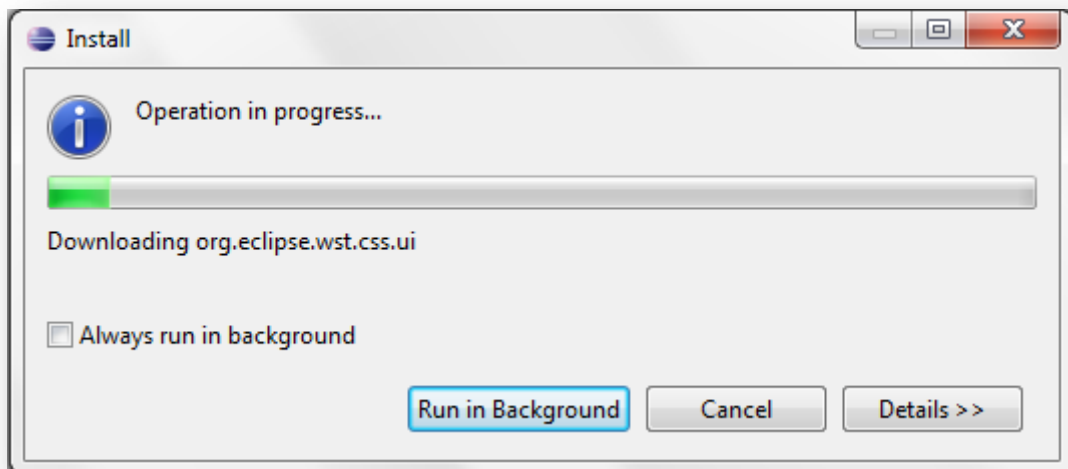
Se nos presentan los detalles de Instalación, para los que nuevamente haremos click sobre “Next”.



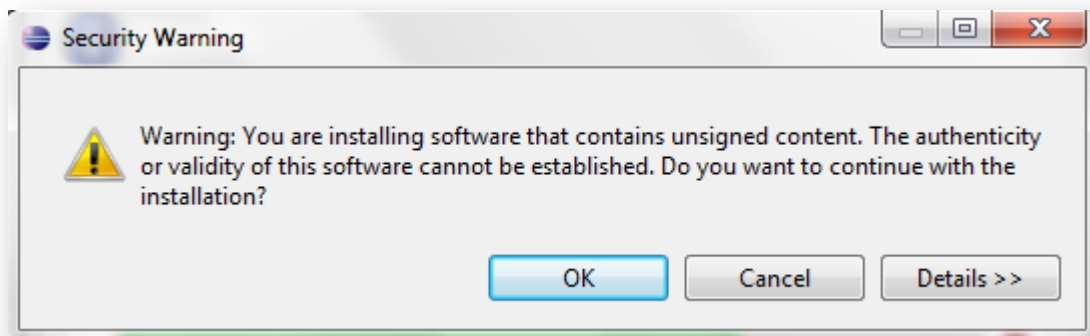
Ahora se nos pregunta si aceptamos los términos de la licencia, para lo que seleccionaremos “I accept the terms of license agreements” y luego haremos click sobre “Finish”.



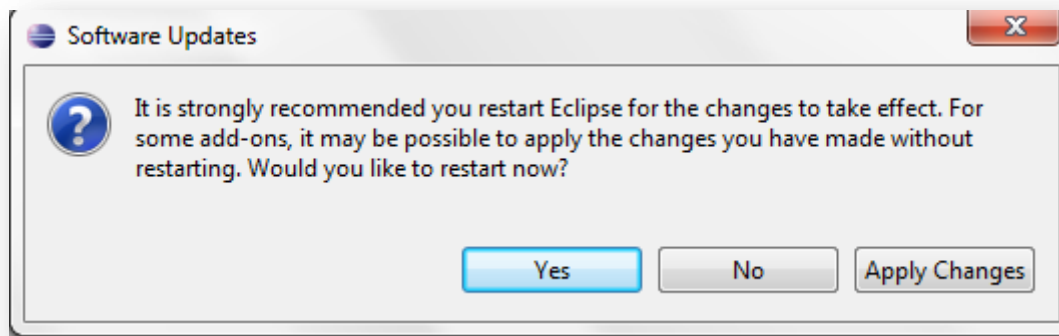
En este momento comenzará a descargarse el Eclipse Google Plugin, por lo que deberemos tener un poco de paciencia hasta que se complete.



Se nos pregunta por la validez del software que estamos instalando, a lo que responderemos haciendo click sobre el botón "Ok".



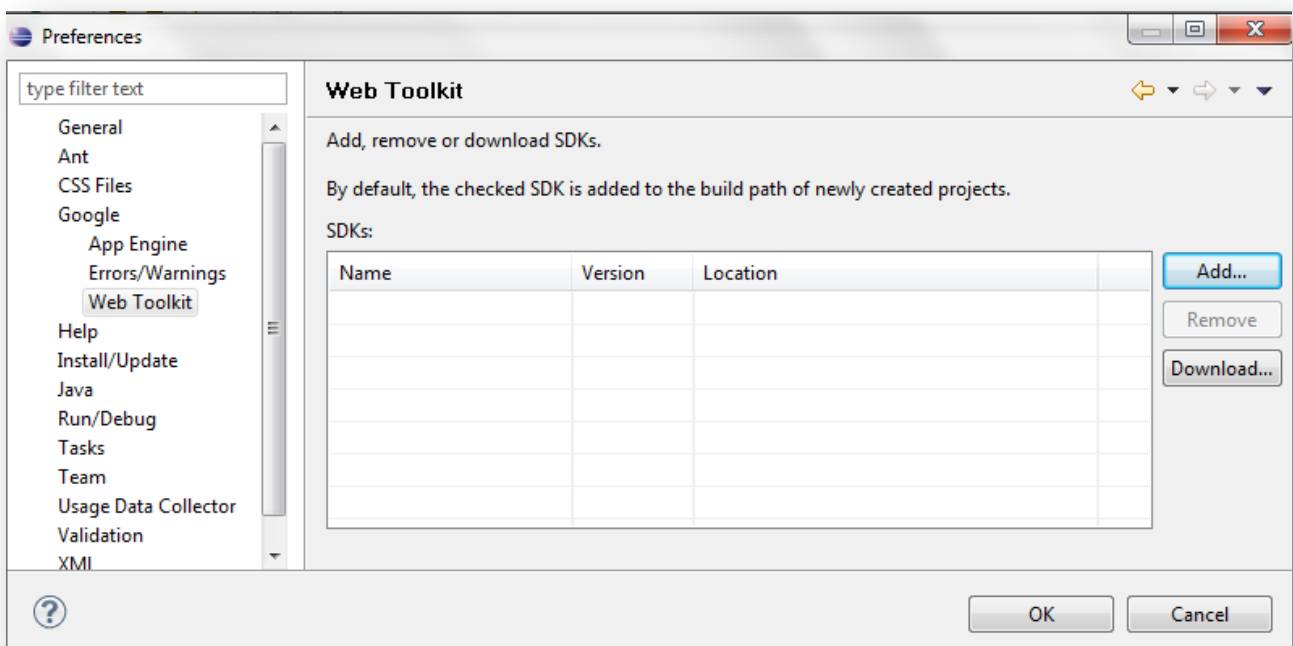
Ahora se nos pregunta si confiamos en el certificado de validez mostrado, por lo que lo seleccionaremos y presionaremos "Ok". Entonces esperaremos a que termine de descargarse el plugin, hasta que Eclipse nos pregunte si deseamos reiniciar el entorno, a lo que responderemos con "Yes".



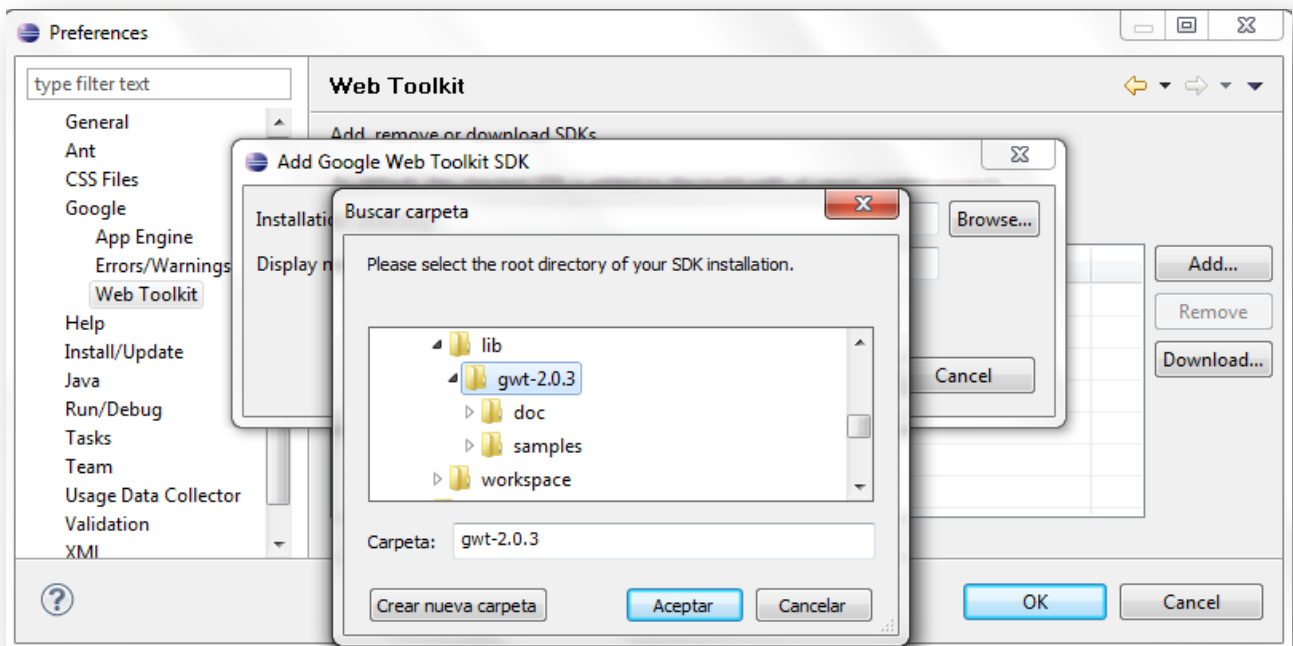
Luego de que Eclipse se reinicie, podremos apreciar la inclusión de 3 botones nuevos en la barra de herramientas. Los dos primeros botones son para uso exclusivo de GWT, como veremos más adelante.



Ahora sólo nos resta configurar el Google Eclipse Plugin para que sepa en dónde se encuentra el SDK de GWT. Para ello iremos al menú "Window->Preferences..." y seleccionaremos la sección de GWT. Allí se nos dará la posibilidad de configurar el SDK, por lo que haremos click en el botón "Add..."

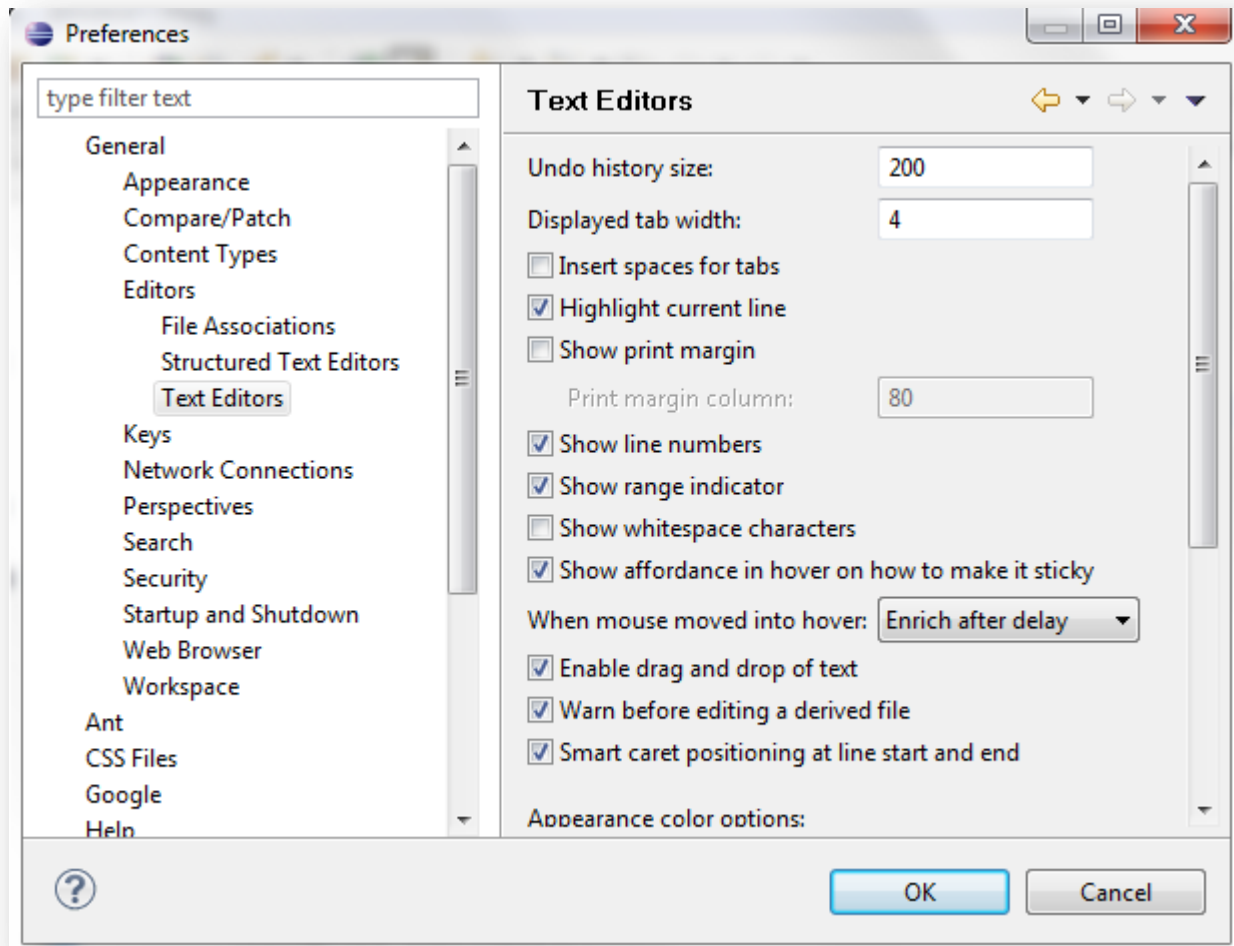


Ahora buscaremos el directorio raíz de nuestra instalación de GWT, que se encuentra dentro de **handsongwt/lib**



Haremos click en “Ok” en los 2 cuadros restantes para finalizar con la instalación de herramientas.

Por último, nos aseguraremos de que los editores de texto de Eclipse nos muestren los números de línea de los archivos fuente. Para ello nos dirigiremos a “Window->Preferences...” y seleccionaremos “Text Editors”, en donde tildaremos la casilla “Show Line Numbers”.



2) Creación del Proyecto Inicial

El Google Eclipse Plugin, provee un asistente para generar proyectos en GWT. Para crear nuestro primer proyecto, haremos click sobre el botón azul con la letra “g” que se encuentra a la izquierda de la nueva barra de herramientas



Esto abre un asistente de creación de proyectos. En la caja de texto “Project Name”, especificaremos el nombre de nuestro proyecto **HolaMundo**, y en “Package” el paquete raíz, que será **com.logikas.eduka.handsongwt.holamundo**. La opción “Use Google App Engine” debe ser desmarcada, ya que no la utilizaremos. Por último haremos click sobre “Finish” para crear el nuevo proyecto.

New Web Application Project

Create a Web Application Project

Create a Web Application project in the workspace or in an external location

Project name:
HolaMundo

Package: (e.g. com.example.myproject)
com.logikas.eduka.handsongwt.holamundo

Location

☒ Create new project in workspace
☐ Create new project in:

Directory: Browse...

Google SDKs

☒ Use Google Web Toolkit

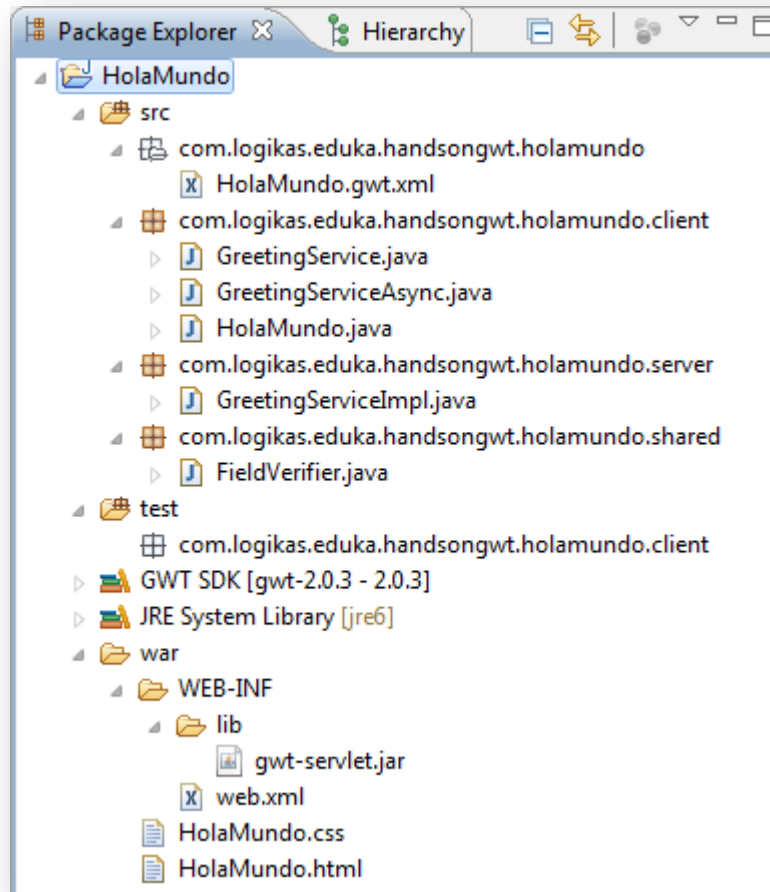
☒ Use default SDK (gwt-2.0.3 - 2.0.3) [Configure SDKs...](#)
☐ Use specific SDK: gwt-2.0.3 - 2.0.3

☐ Use Google App Engine

☒ Use default SDK (none) [Configure SDKs...](#)
☐ Use specific SDK:

[?](#) [Finish](#) [Cancel](#)

El asistente de creación de proyectos nos habrá creado un proyecto con la siguiente estructura:



Más adelante estudiaremos qué significa cada recurso dentro de este proyecto. Por ahora veremos en qué consiste el proyecto creado para lo cual lo haremos correr.

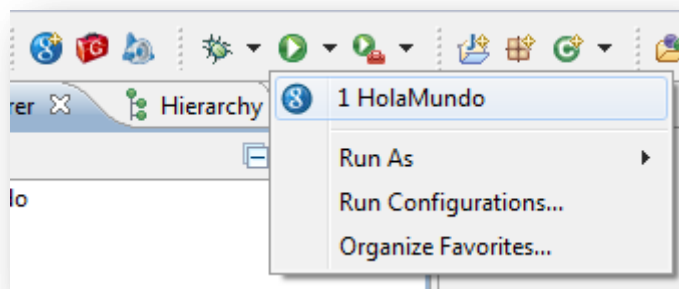
3) Modos de Ejecución de GWT: Development Mode y Web Mode

Es importante en este momento, entender los modos de funcionamiento de GWT. Como dijimos antes, GWT dispone de un compilador que traduce código fuente Java hacia JavaScript. Este compilador realiza un análisis de código muy complejo, por lo que demora mucho tiempo en generar una salida JavaScript compilada. Es por ello que existe algo denominado **Development Mode** (Modo de Desarrollo) que permite ejecutar la aplicación GWT sin necesidad de compilarla, por medio de un intérprete de JavaScript que corre como plugin del navegador. El Development Mode se ejecuta únicamente en fase de desarrollo. Cuando la aplicación se corre en producción, se

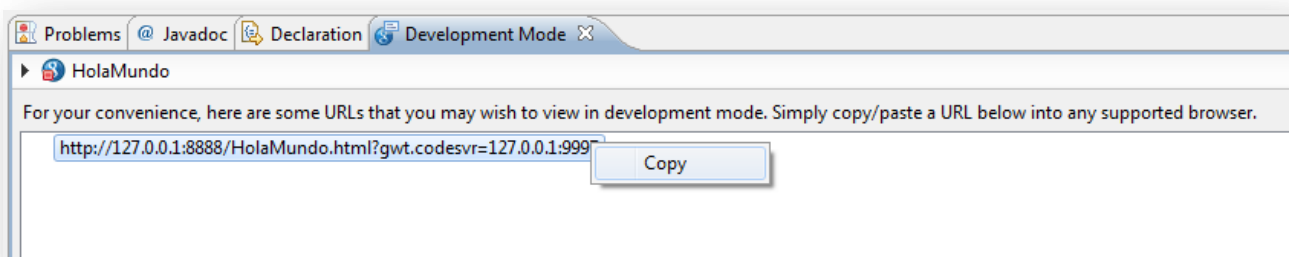


utiliza lo que se conoce como **Web Mode** (Modo Web), que no es más que la aplicación compilada y empaquetada como un WAR y desplegada en un Contenedor de Servlets. La distribución del SDK de GWT trae incluido el Contenedor de Servlets Jetty, que permite realizar rápidas pruebas de funcionamiento sobre la aplicación que estamos desarrollando. En Unidades más avanzadas de este curso, veremos que es posible utilizar diferentes contenedores de servlets y servidores de aplicaciones, como Tomcat, Jboss, Glassfish, etc.. A los fines prácticos de este curso, nos será suficiente el contenedor Jetty.

Para correr el proyecto generado en Development Mode, seleccionaremos la aplicación HolaMundo, en el menú desplegable del botón “Run”.



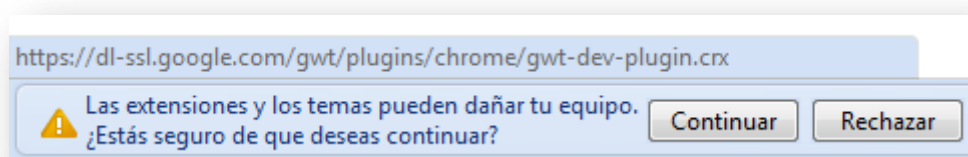
Esta acción habilitará una pestaña en el panel inferior, que mostrará la consola de GWT para **Development Mode**. El texto que encabeza la pestaña, nos sugiere que copiemos la URL mostrada en un navegador. Esta URL apunta hacia nuestra aplicación en **Development Mode**. En este momento ya se encuentra corriendo el contenedor de servlets Jetty, que contiene nuestra aplicación.



Con click derecho sobre la URL, podremos copiarla y luego pegarla sobre el navegador. Para ello iniciaremos Google Chrome, y copiaremos la URL sobre la barra de direcciones. A continuación veremos un mensaje en la ventana del navegador, que nos indica que es necesario instalar el **Plugin para Desarrolladores**, que es necesario para correr la aplicación en Development Mode. Existe un plugin para desarrolladores para cada uno de los navegadores soportados por GWT. Es importante entender que **este plugin es únicamente requerido en fase de desarrollo para correr en Development Mode, y que el usuario final jamás requerirá instalar ninguna extensión en su navegador.**



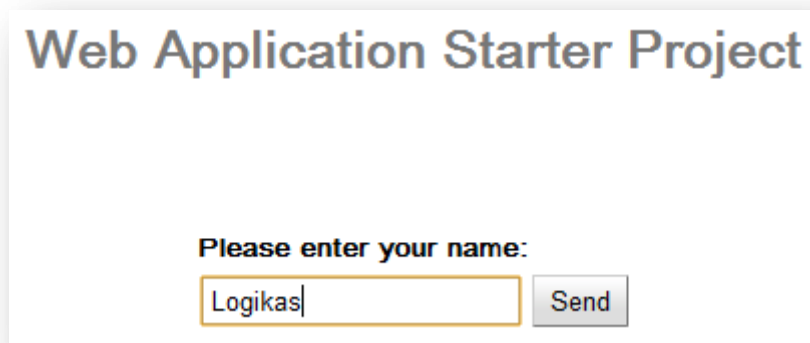
Haremos click sobre el botón azul para instalar el Plugin para Desarrolladores, y esperaremos a que Google Chrome lo descargue. En la esquina inferior izquierda del navegador se nos preguntará si estamos seguros de instalar la extensión, a lo que responderemos con “Continuar”.



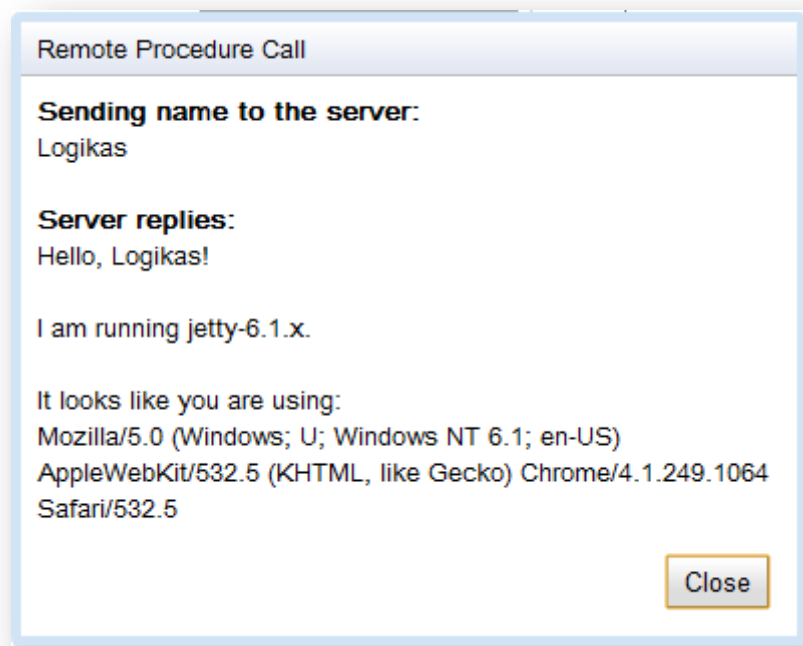
Un último mensaje de confirmación se presenta en la pantalla de Chrome. Haremos click sobre “Instalar” para continuar con la instalación.



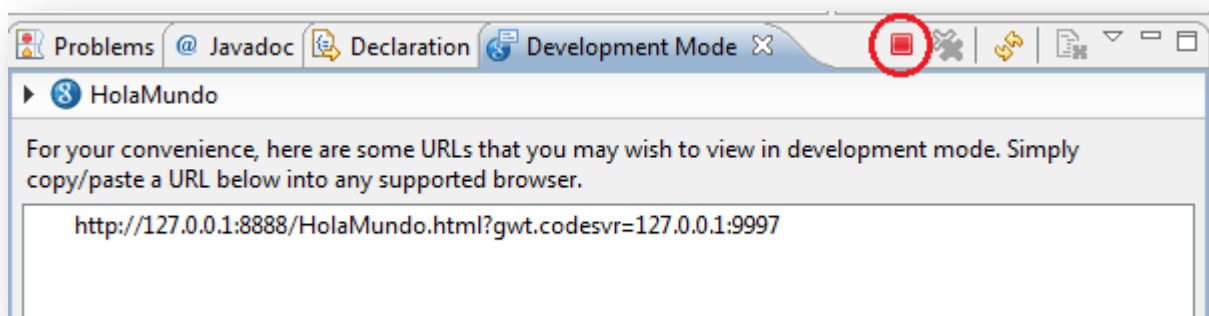
Una vez finalizada la instalación, refrescaremos la página con F5, y veremos que la aplicación web comienza a funcionar, mostrando texto y controles gráficos. Introduciremos nuestro nombre en la caja de texto, y luego haremos click sobre “Send” para conocer qué ocurre.



La aplicación nos devuelve una respuesta, conteniendo el nombre ingresado, y las características de nuestro navegador. Esta respuesta es elaborada en el servidor Jetty, y devuelta por medio de AJAX hacia nuestro navegador.



Para finalizar el servicio de Jetty y el intérprete de Development Mode, haremos click sobre el botón rojo que se encuentra en la esquina superior derecha de la consola de **Development Mode**.

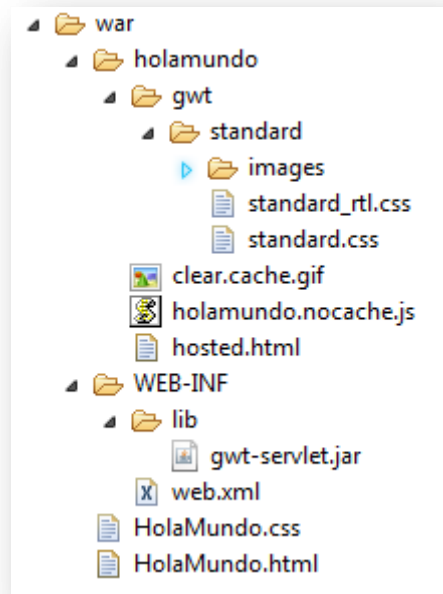


Esta ha sido nuestra primera aplicación GWT. Pronto entenderemos cómo está conformada, y cómo podemos personalizarla.

4) Estructura de un Proyecto GWT

Como vimos anteriormente, el asistente de proyectos de GWT, nos ha generado una particular estructura de proyecto. Dentro de esta estructura, existen diferentes directorios:

4.1) Directorio WAR:



Esta estructura de directorio tiene que resultarnos familiar, ya que conforma la jerarquía clásica de un WAR. En el directorio WEB-INF mantenemos información privada, como las librerías y clases de servidor (inicialmente sólo tenemos la librería de soporte de GWT en el servidor). El directorio WAR contiene la estructura que será desplegada en el servidor. }

4.1.1) Directorio de Aplicación:

Al momento de compilar nuestro proyecto, los archivos de recursos generados (JavaScript, Imágenes, CSS, etc..) serán depositados dentro de un subdirectorio conocido

como “Directorio de Aplicación” que tendrá el mismo nombre que nuestro proyecto pero en minúsculas (en nuestro caso **holamundo**). Para ver este subdirectorio deberemos refrescar el proyecto, ya que en nuestro caso fue generado cuando ejecutamos el proyecto en Development Mode.



4.1.2) Bootstrap File:

Dentro del Directorio de Aplicación existe un archivo muy importante que recibe el nombre de **holamundo.nocache.js**. Este archivo se conoce como **Bootstrap File**, y es el archivo javascript que se incluye en la página principal (Host Page), y tiene como función cargar el resto de la aplicación GWT, como veremos más adelante.

4.1.3) Host Page:

Recordemos que en la introducción de esta unidad comentamos que las aplicaciones GWT son **Single Page**, es porque se ejecutan en una única página HTML que es manipulada enteramente por el navegador. Esta página se conoce como **Host Page**, y generalmente recibe el mismo nombre de archivo que el proyecto. En nuestro caso, la Host Page recibe el nombre de HolaMundo.html. Veremos que el archivo web.xml, indica que HolaMundo.html es la página de inicio de nuestro proyecto, por lo que será la página invocada por defecto cuando se ingrese a la URL raíz de nuestra aplicación. La Host Page viene acompañada por un archivo de estilo CSS, que en nuestro caso recibe el nombre de HolaMundo.css. Este archivo contiene las reglas de estilo por defecto de la Host Page. Más adelante veremos que los estilos de la aplicación son en realidad embebidos de maneras más ingeniosas que esta.

Si revisamos el código de HolaMundo.html, veremos que se trata de una página HTML “pura”, sin ningún tipo de tags o anotaciones especiales, a diferencia de lo que ocurriría en una página JSP.

La etiqueta más importante de este archivo es la que incluye el Bootstrap File, dentro del head de la página (línea 27):

```
<script type="text/javascript" language="javascript" src="holamundo/holamundo.nocache.js"></script>
```

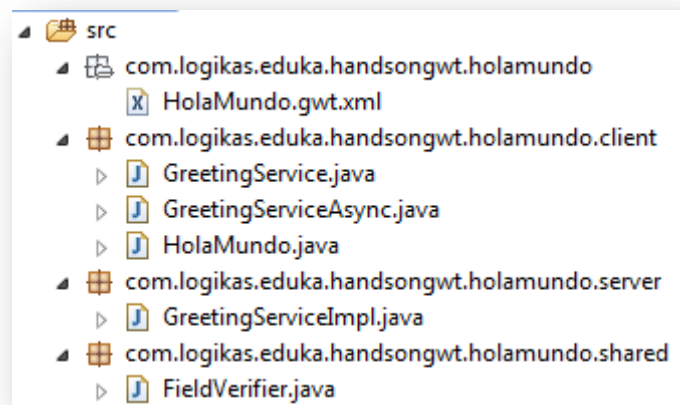
Más abajo veremos un texto conocido (línea 48), encerrado entre etiquetas `<h1>` que corresponden al título de la página que vimos cuando corrimos por primera vez la aplicación generada. Por debajo de este título, nos encontramos con una tabla HTML (línea 50) que contiene



celdas sobre las que se renderizarán los componentes gráficos de GWT.

4.2) Directorio src:

Este directorio, como es habitual, contiene los archivos fuente de Java. Es reconocido por Eclipse como Raíz de todos los paquetes, y por eso se muestra de manera diferente. Dentro de este directorio encontramos los siguientes componentes:



4.2.1) Archivo de Módulo:

Un módulo es la unidad principal de encapsulamiento en GWT. El módulo permite establecer tanto el contenido de una librería GWT como el contenido de una aplicación principal. Los módulos se definen por medio de un archivo XML ubicado dentro del paquete raíz de la aplicación. Cada módulo puede utilizar las funcionalidades de otros módulos, en lo que se conoce como Herencia de Módulos.

En nuestro ejemplo, el módulo principal se llama holamundo y está definido en el archivo HolaMundo.gwt.xml, dentro del paquete **com.logikas.eduka.handsongwt.holamundo**. Este archivo contiene diferentes etiquetas, que permiten definir nuestra aplicación:



4.2.1.1) Etiqueta <module>:

Es la etiqueta raíz del documento XML. Por default, el módulo recibe el mismo nombre del archivo XML que lo define, sin embargo, si en la etiqueta <module> se especifica el atributo `rename-to`, se consigue especificar un nombre diferente, como sucede en el caso de nuestra aplicación con la etiqueta `<module rename-to='holamundo'>...</module>` (línea 2)

4.2.1.2) Etiqueta <inherits>:

Establece qué módulos serán heredados por el módulo actual. En nuestro ejemplo, la etiqueta `<inherits name='com.google.gwt.user.User'>` (línea 4) establece que nuestro módulo incluirá las funcionalidades definidas en el módulo `com.google.gwt.user.User`. El módulo `User` viene predefinido por GWT, y contiene los paquetes principales para desarrollar código en el cliente.

4.2.1.3) Etiqueta <source>:

Define cuáles serán los directorios que el compilador de GWT explorará para compilar clases que se utilizarán en la aplicación cliente. Esto es necesario debido a que sólo un subconjunto de clases Java pueden ser compiladas.

En nuestro ejemplo encontramos las etiquetas `<source path='client'>` y `<source path='shared'>` (líneas 19 y 20), que indican que el compilador explorará los archivos fuente contenidos en los paquetes `com.logikas.eduka.handsongwt.holamundo.client` y `com.logikas.eduka.handsongwt.holamundo.shared` para generar la salida JavaScript.

4.2.1.4) Etiqueta <entry-point>:

Este es un elemento central en una aplicación GWT, ya que define lo que se conoce como **EntryPoint**. `EntryPoint` es una interface que contiene un único método, `onModuleLoad()` que es el equivalente del método `main()` en una aplicación Java clásica. Este es el primer método que se ejecuta luego de que se completa la carga del módulo, y será el responsable de invocar al resto de métodos e instancias que se encargarán de construir la vista. Por tanto, la etiqueta `entry-point`



contiene un atributo `class`, que indica qué clase implementando la interface `EntryPoint` debe ser invocada luego de la carga del módulo. En nuestra aplicación, la etiqueta indica que nuestro `EntryPoint` es una clase llamada `HolaMundo`

```
<entry-point class='com.logikas.eduka.handsongwt.holamundo.client.HolaMundo' /> (línea 16)
```

Los archivos de módulo soportan muchas más etiquetas que las que analizamos, pero por ahora es suficiente con este pequeño conjunto.

4.2.2) Paquetes:

Por convención las aplicaciones GWT se estructuran sobre tres tipos de paquetes: `client`, `shared` y `server`.

4.2.2.1) Paquete `client`:

Contiene el código que se ejecuta únicamente en el navegador, y que debe ser traducido por el compilador hacia JavaScript. Este paquete hace referencia a todas las clases relacionadas con la interfaz de usuario. Es aquí en donde se encuentra la implementación de `EntryPoint` que se utiliza como punto de acceso a la aplicación GWT. Las clases cliente sólo pueden utilizar un conjunto reducido de las clases del JRE conocido como **JRE Emulation**. Esto se debe a que JavaScript no soporta las mismas características que una máquina virtual de Java, como podría ser el acceso a disco. Por otra parte, JavaScript es un entorno muy limitado en recursos de memoria, por lo que emular el API completa de reflection de Java sería insostenible en una aplicación GWT. Es así que existe un conjunto de clases del JRE soportadas por GWT. Estas clases no siempre implementan la completitud de sus métodos, si no sólo los que pueden ser emulados en un cliente JavaScript. El [listado completo de las clases emuladas](#) se puede obtener en el sitio de GWT.

4.2.2.2) Paquete `shared`:

Agrupar las clases que pueden ser utilizadas tanto en el cliente como en el servidor, por lo que también deben compilarse hacia JavaScript. Generalmente agrupa los objetos de transferencia de datos, que se intercambian durante la comunicación AJAX entre el navegador y el servidor.



Presenta las mismas limitaciones que el paquete client en cuanto a utilización de clases del JRE.

4.2.2.3) Paquete server:

Contiene las clases que se ejecutarán únicamente en el servidor, por lo cual sus archivos fuente no interesan al compilador de GWT. Estas clases pueden utilizar el lenguaje Java en toda su extensión, sin restricciones sobre el JRE.

Debemos notar que los directorios correspondientes a los paquetes **client** y **shared** fueron referenciados con la etiqueta <source> en el archivo **HolaMundo.gwt.xml**. Esto se debe a que el compilador de GWT requiere leer el código fuente de ambos paquetes para poder generar un equivalente en JavaScript. Esto no es necesario en el paquete server, ya que sus clases se ejecutan únicamente en el servidor.

5) Definición de la Interfaz de Usuario:

En nuestro ejemplo, el asistente de creación de proyectos ha generado los paquetes client, shared y server con clases de ejemplo que luego nosotros podremos modificar. Por ahora nos centraremos en el paquete **com.logikas.eduka.handsongwt.holamundo.client** y en la clase **HolaMundo**, que es nuestra implementación de la interface **EntryPoint**, como vemos en la línea 23 del archivo HolaMundo.java

```
public class HolaMundo implements EntryPoint
```

5.1) Widgets:

Como podemos ver, el método **onModuleLoad()** fue implementado por el asistente y contiene una serie de sentencias destinadas a construir la interface gráfica. La primera línea, nos muestra la creación de un botón por medio de la sentencia

```
final Button sendButton = new Button("Send");
```



Esta expresión instancia el botón con el texto “Send” que anteriormente clickeamos cuando hicimos correr la aplicación. En la línea siguiente (línea 43), encontramos una instanciación de la clase `TextBox`, que representa la caja de texto sobre la que cargamos nuestro nombre durante la prueba.

```
final TextBox nameField = new TextBox();
```

En GWT, los componentes gráficos reciben el nombre de **Widgets** y descienden de la clase `com.google.gwt.user.client.ui.Widget`. Cada widget contiene propiedades con las que se interactúa por medio de getters y setters, como por ejemplo alto, ancho, estilo CSS, texto etc.. En la línea 44 vemos cómo se establece la propiedad “text” en el `TextBox` instanciado anteriormente.

```
nameField.setText("GWT User");
```

Algunos widgets tienen la capacidad de contener otros widgets en su interior, por lo que una aplicación GWT se conforma por árboles de widgets incrustados en la Host Page. La clase `RootPanel` permite, por medio del atributo `id`, insertar widgets dentro de los contenedores HTML (`div`, `td`, `th`, etc.) que se encuentran en la Host Page. La sentencia que se encuentra en la línea 53...

```
RootPanel.get("sendButtonContainer").add(sendButton);
```

...muestra como insertar el botón `sendButton` en la celda con `id` “`sendButtonContainer`” que se encuentra definida en la línea 56 del archivo `HolaMundo.html`.

```
<td id="sendButtonContainer"></td>
```

5.2) Eventos:

Los widgets también son fuente de eventos de usuario, como click, selección, enfoque, etc., que pueden ser escuchados por manejadores de evento. Los manejadores de evento implementan interfaces que generalmente contienen un único método, que es invocado cuando el evento es disparado. En la línea 80 podemos ver cómo se define un manejador para el evento click.



En este manejador, el método `onClick()` es invocado cuando el usuario hace click sobre el botón de cierre del cuadro de diálogo.

```
closeButton.addClickHandler(new ClickHandler() {  
    public void onClick(ClickEvent event) {  
        dialogBox.hide();  
        sendButton.setEnabled(true);  
        sendButton.setFocus(true);  
    }  
});
```

6) Capacidades del Development Mode:

Ahora analizaremos algunas de las características que nos ofrece el **Development Mode**. Para ello iniciaremos el entorno de Development Mode en la manera que vimos anteriormente.

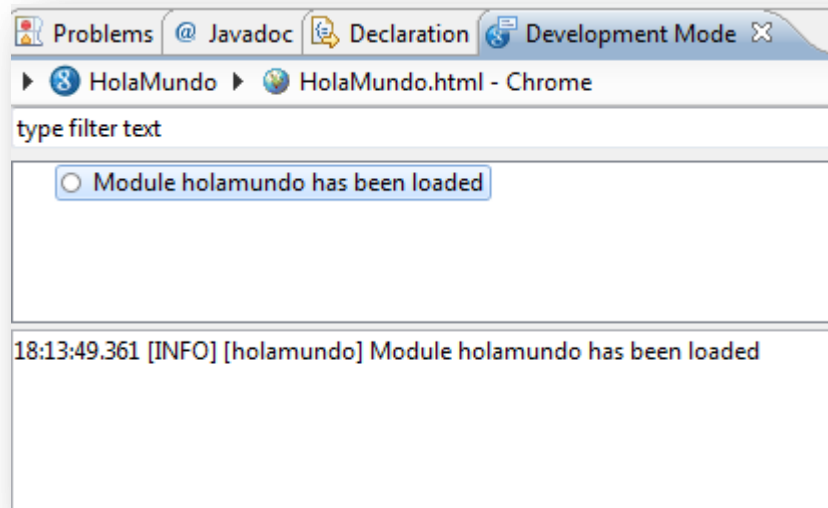
6.1) Reemplazo de código en caliente:

El Development Mode permite modificar el código cliente de la aplicación sin necesidad de redespugarla. Para comprobarlo, modificaremos el String “Send” de la línea 42 de `HolaMundo.java` por el String “Enviar”, y luego actualizaremos la página presionando F5 en el navegador para ver cómo ha cambiado el texto.



6.2) Consola

La consola muestra mensajes de información y de error, que se irán disparando durante la ejecución de la aplicación. Todas las excepciones no atrapadas que ocurran en el cliente, serán reflejadas por esta consola.



6.3) Depuración:

Una de las características de GWT que marca enorme diferencia con los toolkits de JavaScript clásicos es la capacidad de depuración que posee. Para iniciar el depurador, primero cerraremos la consola actual del Development Mode, para luego iniciarla en modo de depuración haciendo click sobre el ícono de debug.



Ahora estableceremos un punto de depuración en la línea 94, que se ejecuta cuando el usuario hace click sobre el botón de “Enviar”

```
93 public void onClick(ClickEvent event) {  
94     sendNameToServer();  
95 }
```

Sobre la página web haremos click en “Enviar”. Si maximizamos la ventana de Eclipse, veremos que se nos pregunta si deseamos cambiar a la perspectiva de Debug, a lo que responderemos con “Yes”. Luego el depurador nos marcará la línea sobre la que marcamos el punto de depuración.

```
93 public void onClick(ClickEvent event) {  
94     sendNameToServer();  
95 }
```

Esto nos demuestra que el Development Mode nos permite depurar una aplicación GWT de la misma forma en que lo haríamos con una aplicación Java común y corriente. Para finalizar la ejecución del evento, haremos click sobre el botón de “Resume”.



7) Compilación y Web Mode:

En esta sección veremos cómo se comporta la aplicación corriendo en JavaScript puro, sin intérpretes intermedios, entendiendo en qué consiste el proceso de compilación de GWT.

7.1) Archivos de Aplicación:

Por cada uno de los navegadores soportados GWT genera un archivo JavaScript diferente, que contiene una versión optimizada de la aplicación y se conoce como Archivo de Aplicación. En los toolkits clásicos de JavaScript como jQuery y Prototype esta característica es inexistente, por lo que el código JavaScript debe determinar en cada momento cómo ejecutar ciertas funcionalidades según el navegador sobre el que se encuentre corriendo, agregando retrasos inútiles a la ejecución de las aplicaciones.



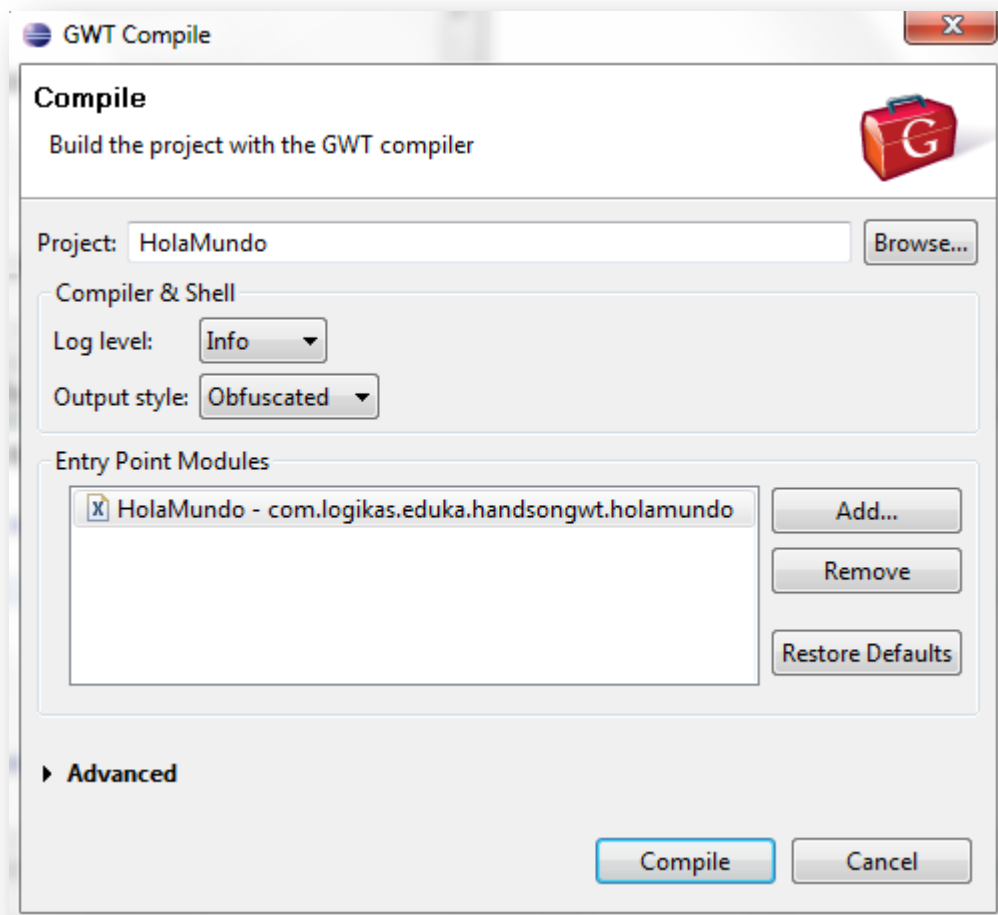
El ciclo de compilación que genera cada Application File en GWT recibe el nombre de Permutación.

Los archivos de aplicación, son cargados por el archivo de bootstrap cuando la página es cargada. El archivo de bootstrap decide qué archivo de aplicación cargar, según el navegador en el que se encuentre corriendo. Los archivos de aplicación reciben por nombre un código MD5 seguido por la extensión “.cache.html”.

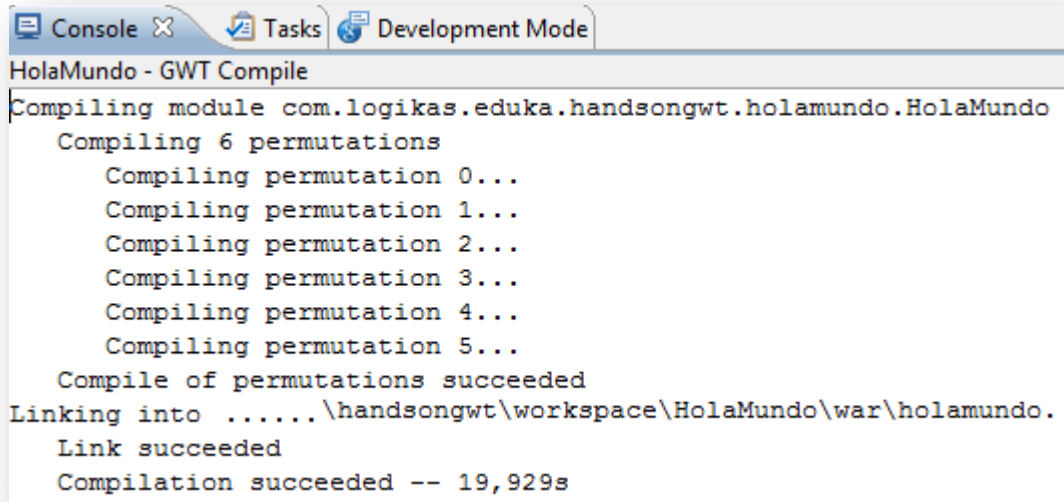
Ahora compilaremos nuestro proyecto para analizar los archivos que genera el compilador. Para ello haremos click sobre el botón de compilación de GWT.



Esta acción desplegará un cuadro de diálogo que nos permite establecer varios detalles de compilación. Por ahora sólo nos interesa la lista que muestra los módulos que contienen los EntryPoint, en nuestro caso únicamente holamundo.



Haremos click sobre "Compile" y observaremos la consola del compilador.

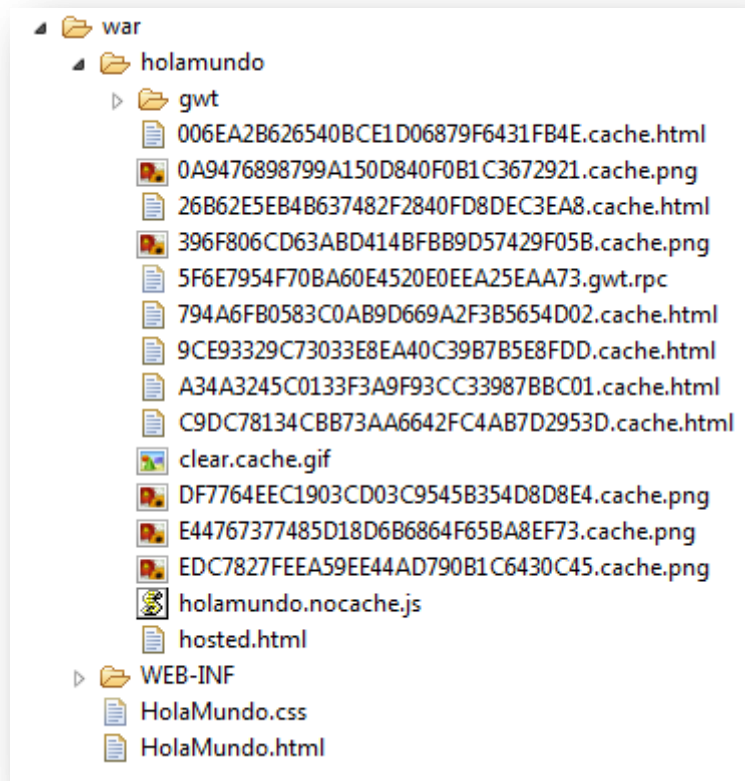


```
Console X Tasks Development Mode
HolaMundo - GWT Compile
Compiling module com.logikas.eduka.handsongwt.holamundo.HolaMundo
  Compiling 6 permutations
    Compiling permutation 0...
    Compiling permutation 1...
    Compiling permutation 2...
    Compiling permutation 3...
    Compiling permutation 4...
    Compiling permutation 5...
  Compile of permutations succeeded
Linking into .....\\handsongwt\\workspace\\HolaMundo\\war\\holamundo.
Link succeeded
Compilation succeeded -- 19,929s
```

La consola nos muestra que se han realizado 6 permutaciones, es decir que tenemos 6 archivos de aplicación diferentes de JavaScript, uno por cada navegador soportado.

La consola nos informa además, que el proyecto fue linkeado en el directorio handsongwt/workspace/HolaMundo/war/holamundo. Esto significa que los archivos generados por el compilador, fueron depositados en ese directorio para ser desplegados en el contenedor de servlets.

Refrescando nuestro proyecto en Eclipse, veremos que el directorio WAR contiene una serie de archivos nuevos.



Prestemos atención a los archivos de extensión “.cache.html”. Efectivamente son 6, que es la misma cantidad de permutaciones que nos informó la consola del compilador.

Un detalle que llama la atención es que los archivos de aplicación tengan extensión “.cache.html”, mientras que el archivo de bootstrap tiene extensión “.**nocache**.js”. ¿Qué significa eso de “**cache**” y “**nocache**”? Como su nombre lo indica, los archivos con extensión “cache” pueden ser cacheados en el navegador por tiempo ilimitado, mientras que los archivos “nocache” nunca deberían ser cacheados. Es posible establecer que el servidor indique en los headers HTTP la duración que debe tener cada recurso en el navegador. En el caso de los archivos “cache”, se guardan “por siempre”. Ahora cabe preguntarse cómo se entera el navegador el hecho de que nosotros cambiemos el código de nuestra aplicación y la volvamos a desplegar. ¿El navegador seguirá mostrando la aplicación vieja? La respuesta es que no, porque los nombres MD5 de los archivos cache son generados a partir de aplicar una suma de verificación sobre el código fuente de nuestra aplicación. Si agregamos una línea en nuestro código, la suma de verificación cambiará, y por tanto el nombre de archivo será diferente.

7.2) Web Mode:

Para concluir con la primera Unidad del Curso, correremos nuestra aplicación en Web Mode, sin plugins de navegador intermedios. Para ello simplemente haremos click sobre el botón de ejecución, y colocaremos sobre nuestro navegador la URL que nos devuelve la consola del Development Mode, pero quitando todo el código que comienza desde el signo de pregunta. Esto quiere decir, que en lugar de `http://127.0.0.1:8888/HolaMundo.html?gwt.codesvr=127.0.0.1:9997`, colocaremos simplemente `http://127.0.0.1:8888/HolaMundo.html`.

Veremos que la página se carga de forma mucho más veloz que en Development

Mode. Esto se debe a que está corriendo en JavaScript nativo y optimizado, y así es como correrán las aplicaciones en el mundo real.

Conclusión:

En esta unidad hemos tenido nuestro primer contacto con GWT creando una aplicación básica, conociendo la estructura de proyectos, los distintos modos de ejecución, los componentes gráficos y el manejo de eventos. En las futuras unidades profundizaremos en cada uno de estos temas para producir una aplicación del mundo real.

Enlaces de Interés:

- Sitio de GWT: <http://code.google.com>
- Blog oficial de GWT: <http://googlewebtoolkit.blogspot.com/>
- Foro de desarrolladores de GWT: <http://groups.google.com/group/Google-Web-Toolkit>
- OnGWT, sitio de novedades: <http://www.ongwt.com>
- Sitio oficial del Google Eclipse Plugin: <http://code.google.com/eclipse/>

